

MMM	MMM	TTTTTTTTTTTTTTTT	AAAAAAAAA	AAAAAAAAA	CCCCCCCCCCCC	PPPPPPPPPPPP	
MMM	MMM	TTTTTTTTTTTTTTTT	AAAAAAAAA	AAAAAAAAA	CCCCCCCCCCCC	PPPPPPPPPPPP	
MMM	MMM	TTTTTTTTTTTTTTTT	AAAAAAAAA	AAAAAAAAA	CCCCCCCCCCCC	PPPPPPPPPPPP	
MMMMMM	MMMMMM	TTT	AAA	AAA	CCC	PPP	PPP
MMMMMM	MMMMMM	TTT	AAA	AAA	CCC	PPP	PPP
MMMMMM	MMMMMM	TTT	AAA	AAA	CCC	PPP	PPP
MMM	MMM	TTT	AAA	AAA	CCC	PPP	PPP
MMM	MMM	TTT	AAA	AAA	CCC	PPP	PPP
MMM	MMM	TTT	AAA	AAA	CCC	PPP	PPP
MMM	MMM	TTT	AAA	AAA	CCC	PPP	PPP
MMM	MMM	TTT	AAA	AAA	CCC	PPPPPPPPPPPP	
MMM	MMM	TTT	AAA	AAA	CCC	PPPPPPPPPPPP	
MMM	MMM	TTT	AAA	AAA	CCC	PPPPPPPPPPPP	
MMM	MMM	TTT	AAAAAAAAAAAAAAAA	AAAAAAAAAAAAAAAA	CCC	PPP	
MMM	MMM	TTT	AAAAAAAAAAAAAAAA	AAAAAAAAAAAAAAAA	CCC	PPP	
MMM	MMM	TTT	AAAAAAAAAAAAAAAA	AAAAAAAAAAAAAAAA	CCC	PPP	
MMM	MMM	TTT	AAA	AAA	CCC	PPP	
MMM	MMM	TTT	AAA	AAA	CCC	PPP	
MMM	MMM	TTT	AAA	AAA	CCC	PPP	
MMM	MMM	TTT	AAA	AAA	CCC	PPP	
MMM	MMM	TTT	AAA	AAA	CCC	PPP	
MMM	MMM	TTT	AAA	AAA	CCC	PPP	
MMM	MMM	TTT	AAA	AAA	CCCCCCCCCCCC	PPP	
MMM	MMM	TTT	AAA	AAA	CCCCCCCCCCCC	PPP	
MMM	MMM	TTT	AAA	AAA	CCCCCCCCCCCC	PPP	

```

NN      NN  XX      XX      TTTT TTTT TTTT  VV      VV      000000  LL
NN      NN  XX      XX      TTTT TTTT TTTT  VV      VV      000000  LL
NN      NN  XX      XX      TT          VV      VV      00      00  LL
NN      NN  XX      XX      TT          VV      VV      00      00  LL
NNNN     NN      XX      XX      TT          VV      VV      00      00  LL
NNNN     NN      XX      XX      TT          VV      VV      00      00  LL
NN  NN  NN      XX      TT          VV      VV      00      00  LL
NN  NN  NN      XX      TT          VV      VV      00      00  LL
NN      NNNN     XX      XX      TT          VV      VV      00      00  LL
NN      NNNN     XX      XX      TT          VV      VV      00      00  LL
NN      NN  XX      XX      TT          VV      VV      00      00  LL
NN      NN  XX      XX      TT          VV      VV      00      00  LL
NN      NN  XX      XX      TT          VV      VV      000000  LLLLLLLLLLLL
NN      NN  XX      XX      TT          VV      VV      000000  LLLLLLLLLLLL

```

```

LL          IIIII
LL          IIIII
    II
LL         II
LL         II
LL         II
LL         II
LL         II
LL         II
LL         II
LL         II
LL         II
LL         II
LLLLLLLLLLL
LLLLLLLLLLL

SSSSSSSSS
SSSSSSSSS
    SS
    SS
    SS
    SS
        SSSSSS
        SSSSSS
                SS
                SS
                SS
                SS
SSSSSSSSS
SSSSSSSSS

```



```
0001 0 MODULE NXTVOL (LANGUAGE (BLISS32) ,
0002 0 IDENT = 'V04-000' ,
0003 0 ) =
0004 1 BEGIN
0005 1
0006 1 *****
0007 1 *
0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0010 1 * ALL RIGHTS RESERVED.
0011 1 *
0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0017 1 * TRANSFERRED.
0018 1 *
0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0021 1 * CORPORATION.
0022 1 *
0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0025 1 *
0026 1 *
0027 1 *****
0028 1
0029 1 ++
0030 1
0031 1 FACILITY: MTAACP
0032 1
0033 1 ABSTRACT:
0034 1 This module gets the next volume for read and write
0035 1
0036 1
0037 1 ENVIRONMENT:
0038 1
0039 1 VMS operating system, including privileged system services
0040 1 and internal exec routines.
0041 1
0042 1 --
0043 1
0044 1
0045 1
0046 1 AUTHOR: D. H. GILLESPIE, CREATION DATE: 20-AUG-1977
0047 1
0048 1 MODIFIED BY:
0049 1
0050 1 V03-010 HH0041 Hai Huang 24-Jul-1984
0051 1 Remove REQUIRE 'LIBD$: [VMSLIB.OBJ]MOUNTMSG.B32'.
0052 1
0053 1 V03-009 MMD0287 Meg Dumont, 10-Apr-1984 14:14
0054 1 Fix to the $MTACCESS return code where the ACCESS field
0055 1 could have gotten set to normal processing before
0056 1 all the errors were checked.
0057 1
```

```
58 0058 1 V03-008 LMP0221 L. Mark Pilant, 28-Mar-1984 14:50
59 0059 1 Change UCSL_OWNUIC to ORBSL_OWNER and UCSW_VPROT to
60 0060 1 ORBSW_PROT.
61 0061 1
62 0062 1 V03-007 MMD0273 Meg Dumont, 23-Mar-1984 9:42
63 0063 1 Change the processing of the accessibility character fields
64 0064 1 in the VOL1 and or HDR1 label to call the installation
65 0065 1 specific accessibility routine. The return from this
66 0066 1 routine determines the users access to the volume and/or file.
67 0067 1
68 0068 1 V03-006 MMD0177 Meg Dumont, 26-May-1983 15:13
69 0069 1 Change VOL1 to indicate ANSI level 4 when writing SYSTEM CODE
70 0070 1 in VOL1 LABEL
71 0071 1
72 0072 1 V03-005 MMD0159 Meg Dumont, 26-Apr-1983 9:30
73 0073 1 Change reference to 240 the symbol SCRATCH_OFFSET.
74 0074 1
75 0075 1 V03-004 MMD0135 Meg Dumont, 12-Apr-1983 17:29
76 0076 1 Added support for writng and interrupting the VOL1
77 0077 1 OWNER IDENTIFIER field, so that it is no longer
78 0078 1 treated as a VMS field, strictly.
79 0079 1
80 0080 1
81 0081 1 V03-003 MMD0121 Meg Dumont, 29-Mar-1983 0:45
82 0082 1 Added support for the VOL2 label inside the MTAACP
83 0083 1
84 0084 1 V03-002 MMD0104 Meg Dumont, 17-Feb-1983 13:19
85 0085 1 Use GET_DEV_NAME for tape units name. Added code for AVR and AVL
86 0086 1
87 0087 1 V02-015 DMW00060 David Michael Walp 7-Dec-1981
88 0088 1 Rename TRANSLATION_TABLE to ANSI_A_GOOD
89 0089 1
90 0090 1 V02-014 DMW00037 David Michael Walp 17-Sep-1981
91 0091 1 Set MVL entry used when GTNEXT_VOL_READ places the label
92 0092 1 in the MVL
93 0093 1
94 0094 1 V02-013 DMW00031 David Michael Walp 18-Aug-1981
95 0095 1 Volume Access project
96 0096 1
97 0097 1 V02-012 DMW00018 David Michael Walp 20-May-1981
98 0098 1 Checks for File-Set-Id changed to look at the MVL rather
99 0099 1 then VCB ( 1st mounted volume label ).
100 0100 1
101 0101 1 V02-011 REFORMAT Maria del C. Nasr 30-Jun-1980
102 0102 1
103 0103 1 A0010 MCN0003 Maria del C. Nasr 15-Oct-1979 9:26
104 0104 1 Add HDR3 processing
105 0105 1
106 0106 1 A0009 ACG0047 Andrew C. Goldstein, 9-Aug-1979 14:17
107 0107 1 Protection check interface changes
108 0108 1
109 0109 1 **
110 0110 1
111 0111 1 LIBRARY 'SYSS$LIBRARY:LIB.L32';
112 0112 1 REQUIRE 'SRC$:MTADEF.B32';
113 0496 1
114 0497 1 LINKAGE
```


NXTVOL
V04-000

L 13
16-Sep-1984 02:27:10 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:46:45 [MTAACP.SRC]NXTVOL.B32;1

Page 3
(1)

```
: 115      0498 1    CHECK_PROT      = JSB (REGISTER = 0, REGISTER = 1, REGISTER = 2)
: 116      0499 1                                : NOTUSED (3, 4, 5, 6, 7, 8, 9, 10, 11),
: 117      0500 1    L$CHECK_HDR      = JSB : GLOBAL (SCRATCH = 9, CURRENT_VCB = 11)
: 118      0501 1                                NOTUSED (7, 8, 10);
: 119      0502 1
: 120      0503 1    FORWARD ROUTINE
: 121      0504 1      CHECK HDR      : L$CHECK_HDR,      ! check that HDR can be overwritten
: 122      0505 1      GTNEXT_VOL_READ : NOVALUE L$GTNEXT_VOL_RE, ! get next volume for read
: 123      0506 1      GTNEXT_VOL_WRIT : NOVALUE L$GTNEXT_VOL_WR, ! get next volume for write
: 124      0507 1      INC_VOL_SECTION : COMMON_CALL NOVALUE,    ! incr rel vol and sect #
: 125      0508 1      RESET_UNIT      : COMMON_CALL NOVALUE;
: 126      0509 1      UPDATE_MVL_LBL  : COMMON_CALL NOVALUE;    ! update label in MVL entry
: 127      0510 1
: 128      0511 1    EXTERNAL
: 129      0512 1      CURRENT_UCB      : REF BBLOCK,      ! addr current unit control block
: 130      0513 1      HDR1            : REF BBLOCK,      ! addr of HDR1(EOF1) label
: 131      0514 1      IO_PACKET       : REF BBLOCK,      ! addr current I/O request packet
: 132      0515 1      SCH$GL_PCBVEC   : REF VECTOR ADDRESSING_MODE (ABSOLUTE),
: 133      0516 1      WORK_AREA;
: 134      0517 1
: 135      0518 1    EXTERNAL ROUTINE
: 136      0519 1      EXPIRED          : COMMON_CALL,      ! determine if file has expired
: 137      0520 1      FORMAT_VOOWNER  : NOVALUE,          ! format the owner field in VOL2
: 138      0521 1      GET_DEV_NAME     : COMMON_CALL NOVALUE, ! given UCB addr get dev name
: 139      0522 1      GET_RECORD,      :                  ! get record tape is currently reading
: 140      0523 1      ISSUE_IO        : L$ISSUE_IO,      ! issue I/O
: 141      0524 1      MOUNT_VOL       : COMMON_CALL,      ! mount relative vol
: 142      0525 1      PRINT_OPR_MSG    : L$PRINT_OPR_MSG,  ! print a system mess for oper
: 143      0526 1      READ_HDR        : COMMON_CALL;      ! read headers
: 144      0527 1      REWIND_AND_WAIT : COMMON_CALL;
```

```
146 0528 1 GLOBAL ROUTINE GTNEXT_VOL_READ : NOVALUE L$GTNEXT_VOL_RE =
147 0529 1
148 0530 1 ++
149 0531 1
150 0532 1 FUNCTIONAL DESCRIPTION:
151 0533 1 This routine gets the next volume for read and checks that the file
152 0534 1 sequence number, file section number and volume set identifier
153 0535 1 are those sought
154 0536 1
155 0537 1 CALLING SEQUENCE:
156 0538 1 GTNEXT_VOL_READ()
157 0539 1
158 0540 1 INPUT PARAMETERS:
159 0541 1 NONE
160 0542 1
161 0543 1 IMPLICIT INPUTS:
162 0544 1 CURRENT_VCB - address of current volume control block
163 0545 1
164 0546 1 OUTPUT PARAMETERS:
165 0547 1 NONE
166 0548 1
167 0549 1 IMPLICIT OUTPUTS:
168 0550 1 next relative volume mounted
169 0551 1
170 0552 1 ROUTINE VALUE:
171 0553 1 NONE
172 0554 1
173 0555 1 SIDE EFFECTS:
174 0556 1 NONE
175 0557 1
176 0558 1 --
177 0559 1
178 0560 2 BEGIN
179 0561 2
180 0562 2 EXTERNAL REGISTER
181 0563 2 COMMON_REG;
182 0564 2
183 0565 2 LOCAL
184 0566 2 CVT_DEVNAM : VECTOR [MAX_DEVNAM_LENGTH,BYTE], ! Converted dev name
185 0567 2 CVT_DEVNAM_LENGTH : BYTE, ! and length of dev name
186 0568 2 VOL[BL] : BBLOCK [6], ! current tape volume label
187 0569 2 FLAGS,
188 0570 2 FID, ! file identifier
189 0571 2 MVL_ENTRY : REF BBLOCK, ! addr of current rel vol entry in MVL
190 0572 2 RVN, ! current relative volume number
191 0573 2 MVL : REF BBLOCK; ! magnetic tape volume list
192 0574 2
193 0575 2 FLAGS = $FIELDMASK(MOVSV REWIND) OR $FIELDMASK(MOVSV CHKIFSPC);
194 0576 2 KERNEL_CALL(INC VOL_SECTION); ! incr sequence # and relative vol #
195 0577 2 FID = .CURRENT_VCB[VCB$$_CUR_FID]; ! pickup current file id
196 0578 2 RVN = .CURRENT_VCB[VCB$$_CUR_RVN]; ! pickup cur relative volume #
197 0579 2
198 0580 2 WHILE 1
199 0581 2 DO
200 0582 2 BEGIN
201 0583 2
202 0584 2 LOCAL
```



```
SCRATCH      : REF BBLOCK;

! mount vol, rewind it, check the label if the operator specifies it
MVL_ENTRY = MOUNT_VOL(.RVN, .FLAGS);

SCRATCH = .HDR1 + SCRATCH_OFFSET;
CH$MOVE(VL1$$_VOLLBL, SCRATCH[VL1$T_VOLLBL], VOLLBL);

IF NOT READ_HDR()
THEN
    BEGIN
        ERR_EXIT(SS$_TAPEPOSLOST);
    END;

! This next call will use the UCB address to get the device's name and
! will fill in the fields with that name and the length of the name.
GET_DEV_NAME(CVT_DEVNAM_LENGTH, CVT_DEVNAM);

! on read the next volume has the same volume set id and the fid of the
! next section for the current file
IF .FID NEQ .CURRENT_VCB[VCB$_CUR_FID]
THEN
    PRINT_OPR_MSG(MOUN$_NOTRELVOL, 0, .CURRENT_VCB[VCB$_CUR_RVN],
                  .CVT_DEVNAM_LENGTH, CVT_DEVNAM)
ELSE
    BEGIN
        ! pickup the addr of the MVL
        MVL = .CURRENT_VCB[VCB$_MVL];
        IF CH$NEQ(MVL$_SET_ID, MVL[MVL$_SET_ID],
                  HD1$_FILESETID, HDR1[HD1$_FILESETID], ' ')
        AND
            ! not override set identifier with privs
            NOT ( .CURRENT_VCB[VCB$_OVRSETID]
                  AND .MVL_ENTRY [MVL$_OVERRIDE])
        THEN
            PRINT_OPR_MSG(MOUN$_NOTVOLSET, 0,
                          .CVT_DEVNAM_LENGTH, CVT_DEVNAM,
                          6, MVL[MVL$_SET_ID])
        ELSE
            EXITLOOP;
    END;

FLAGS = $FIELDMASK(MOUN$_REWIND) + $FIELDMASK(MOUN$_MOUNTERR);
KERNEL_CALL(RESET_UNIT);
END;                                     ! end of while loop
```

```
203 0585 3
204 0586 3
205 0587 3
206 0588 3
207 0589 3
208 0590 3
209 0591 3
210 0592 3
211 0593 3
212 0594 3
213 0595 3
214 0596 4
215 0597 4
216 0598 4
217 0599 4
218 0600 3
219 0601 3
220 0602 3
221 0603 3
222 0604 3
223 0605 3
224 0606 3
225 0607 3
226 0608 3
227 0609 3
228 0610 3
229 0611 3
230 0612 3
231 0613 3
232 0614 4
233 0615 4
234 0616 4
235 0617 4
236 0618 4
237 0619 4
238 0620 4
239 0621 4
240 0622 4
241 0623 4
242 0624 4
243 0625 5
244 0626 5
245 0627 5
246 0628 4
247 0629 4
248 0630 4
249 0631 4
250 0632 4
251 0633 4
252 0634 4
253 0635 4
254 0636 3
255 0637 3
256 0638 3
257 0639 3
258 0640 2
259 0641 2
```

: 260
: 2610642 2
0643 1KERNEL_CALL(UPDATE_MVL_LBL, .MVL_ENTRY, VOLLBL);
END;
! end of routine.TITLE NXTVOL
.IDENT \V04-000\.EXTRN CURRENT_UCB, HDR1
.EXTRN IO_PACKET, SCH\$GL_PCBVEC
.EXTRN WORK_AREA, EXPIRED
.EXTRN FORMAT_VOLOWNER
.EXTRN GET_DEV_NAME, GET_RECORD
.EXTRN ISSUE_ID, MOUNT_VOL
.EXTRN PRINT_OPR_MSG, READ_HDR
.EXTRN REWIND_AND_WAIT
.EXTRN SYS\$CMKRNL

.PSECT \$CODE\$,NOWRT,2

		07FC	8F	BB	00000	GTNEXT_VOL_READ::		
			5E	1C	C2	00004	PUSHR	#^M<R2,R3,R4,R5,R6,R7,R8,R9,R10>
			58	05	D0	00007	SUBL2	#28, SP
				7E	D4	0000A	MOVL	#5, FLAGS
				5E	DD	0000C	CLRL	-(SP)
				CF	9F	0000E	PUSHL	SP
		0000V		03	FB	00012	PUSHAB	INC_VOL_SECTION
			9F	AB	D0	00019	CALLS	#3, @SYS\$CMKRNL
			5A	AB	9A	0001D	MOVL	36(CURRENT_VCB), FID
			59	58	DD	00021	MOVZBL	47(CURRENT_VCB), RVN
				59	DD	00023	PUSHL	FLAGS
				02	FB	00025	PUSHL	RVN
				50	D0	0002A	CALLS	#2, MOUNT_VOL
				8F	C1	0002D	MOVL	R0, MVL_ENTRY
				06	28	00037	ADDL3	#320, HDR1, SCRATCH
				00	FB	0003D	MOVCL3	#6, 4(SCRATCH), VOLLBL
				50	E8	00042	CALLS	#0, READ_HDR
				8F	BF	00045	BLBS	R0, 2\$
				AE	9F	00049	CHMU	#548
				AE	9F	0004C	PUSHAB	CVT_DEVNAM
				02	FB	0004F	PUSHAB	CVT_DEVNAM_LENGTH
				5A	D1	00054	CALLS	#2, GET_DEV_NAME
				1B	13	00058	CMPL	FID, 36(CURRENT_VCB)
				AE	9F	0005A	BEQL	3\$
				AE	9A	0005D	PUSHAB	CVT_DEVNAM
				AB	9A	00061	MOVZBL	CVT_DEVNAM_LENGTH, -(SP)
				7E	D4	00065	MOVZBL	47(CURRENT_VCB), -(SP)
				8F	DD	00067	CLRL	-(SP)
				30	0006D	PUSHL	#7504148	
				14	C0	00070	BSBW	PRINT_OPR_MSG
				35	11	00073	ADDL2	#20, SP
				AB	D0	00075	BRB	5\$
				CF	D0	00079	MOVL	52(CURRENT_VCB), MVL
				06	29	0007E	MOVL	HDR1, R0
				39	13	00084	CMPC3	#6, 12(MVL), 21(R0)
				03	E1	00086	BEQL	6\$
				02	E0	0008B	BBC	#3, 44(CURRENT_VCB), 4\$
							BBS	#2, 7(MVL_ENTRY), 6\$

0528
0575
0576
0577
0578
0589
0591
0592
0594
0597
0603
0608
0610
0611
0610
0618
0620
0625
0626

NXTVOL
V04-000

C 14
16-Sep-1984 02:27:10
14-Sep-1984 12:46:45

VAX-11 Bliss-32 V4.0-742
[MTAACP.SRC]NXTVOL.B32;1

Page 7
(2)

		0C	A6	9F	00090	4\$:	PUSHAB	12(MVL)	:	0631
			06	DD	00093		PUSHL	#6	:	
		14	AE	9F	00095		PUSHAB	CVT_DEVNAM	:	0629
7E		0C	AE	9A	00098		MOVZBL	CVT_DEVNAM_LENGTH, -(SP)	:	0631
			7E	D4	0009C		CLRL	-(SP)	:	
		0072810C	8F	DD	0009E		PUSHL	#7504140	:	
			0000G	30	000A4		BSBW	PRINT_OPR_MSG	:	
5E			18	C0	000A7		ADDL2	#24, SP	:	
58			09	D0	000AA	5\$:	MOVL	#9, FLAGS	:	0638
			7E	D4	000AD		CLRL	-(SP)	:	0639
			5E	DD	000AF		PUSHL	SP	:	
		0000V	CF	9F	000B1		PUSHAB	RESET UNIT	:	
00000000G	9F		03	FB	000B5		CALLS	#3, @SYSSCMKRN	:	
			FF62	31	000BC		BRW	1\$:	0580
		04	AE	9F	000BF	6\$:	PUSHAB	VOLLBL	:	0642
			57	DD	000C2		PUSHL	MVL_ENTRY	:	
			02	DD	000C4		PUSHL	#2	:	
			5E	DD	000C6		PUSHL	SP	:	
		0000V	CF	9F	000C8		PUSHAB	UPDATE_MVL_LBL	:	
00000000G	9F		05	FB	000CC		CALLS	#5, @SYSSCMKRN	:	
	5E		1C	C0	000D3		ADDL2	#28, SP	:	0643
		07FC	8F	BA	000D6		POPR	#^M<R2,R3,R4,R5,R6,R7,R8,R9,R10>	:	
			05	000DA			RSB		:	

; Routine Size: 219 bytes, Routine Base: \$CODE\$ + 0000

; 262 0644 1

```
264 0645 1 GLOBAL ROUTINE GTNEXT_VOL_WRIT : NOVALUE L$GTNEXT_VOL_WR =
265 0646 1
266 0647 1 ++
267 0648 1
268 0649 1 FUNCTIONAL DESCRIPTION:
269 0650 1 This routine gets the next volume for write. The volume
270 0651 1 is mounted, rewound and the label is verified. The VOL1
271 0652 1 label is rewritten to insure same density throughout volume set.
272 0653 1 The tape is initialized at the operator's request. The tape is
273 0654 1 also inited at the request of the user who mounted the tape.
274 0655 1 That is if the tape was mounted /INIT or /BLANK then every new
275 0656 1 reel in the volume set will be inited if the user has the proper
276 0657 1 access to the tape.
277 0658 1
278 0659 1 CALLING SEQUENCE:
279 0660 1 GTNEXT_VOL_WRIT()
280 0661 1
281 0662 1 INPUT PARAMETERS:
282 0663 1 NONE
283 0664 1
284 0665 1 IMPLICIT INPUTS:
285 0666 1 CURRENT_UCB - address of current unit control block
286 0667 1 CURRENT_VCB - address of current volume control block
287 0668 1 operator input
288 0669 1
289 0670 1 OUTPUT PARAMETERS:
290 0671 1 NONE
291 0672 1
292 0673 1 IMPLICIT OUTPUTS:
293 0674 1 relative volume number incremented
294 0675 1 section number increment
295 0676 1
296 0677 1 ROUTINE VALUE:
297 0678 1 NONE
298 0679 1
299 0680 1 SIDE EFFECTS:
300 0681 1 NONE
301 0682 1
302 0683 1 --
303 0684 1
304 0685 2 BEGIN
305 0686 2
306 0687 2 LITERAL
307 0688 2 BLANK = 0,
308 0689 2 INIT = 1;
309 0690 2
310 0691 2 LOCAL
311 0692 2 CHAR : VECTOR [4,BYTE], ! Char to write in accessibility field
312 0693 2 CURRENT_RECORD, ! current record tape drive is reading
313 0694 2 CVT_DEVNAM : VECTOR [MAX_DEVNAM_LENGTH,BYTE], ! Converted dev name
314 0695 2 CVT_DEVNAM_LENGTH : BYTE, ! and length of dev name
315 0696 2 ERROR_NO,
316 0697 2 FLAGS,
317 0698 2 ORB : REF BBLOCK, ! ORB address
318 0699 2 MVL : REF BBLOCK, ! MVL of current volume set
319 0700 2 MVL_ENTRY : REF BBLOCK, ! Entry of current volume
320 0701 2 SAVE_DEVCHAR : VECTOR [2],
```



```

321 0702 2      OPR_FLAG      : BITVECTOR [2],
322 0703 2      ACCESS_CHAR   : BYTE,
323 0704 2      VOL_OWNER     : VECTOR [ VL1$$ OWNER_IDENT, BYTE],
324 0705 2      SCRATCH2      : BBLOCK [ANSI_LBLSZ],
325 0706 2      STATUS;
326 0707 2
327 0708 2      GLOBAL REGISTER
328 0709 2      SCRATCH = 9      : REF BBLOCK;
329 0710 2
330 0711 2      BIND
331 0712 2      MAIL = WORK_AREA      : BBLOCK [MSGSIZE],
332 0713 2      MAILSZ = MAIL + MSGSIZE,
333 0714 2      STARID = UPLIT ('DECFILÉ11A');
334 0715 2
335 0716 2      EXTERNAL REGISTER
336 0717 2      COMMON_REG;
337 0718 2
338 0719 2      KERNEL CALL(INC_VOL_SECTION);
339 0720 2      SAVE_DEVCHAR[0] = .CURRENT_UCB[UCB$B_DEVCLASS]<0, 32>;
340 0721 2      SAVE_DEVCHAR[1] = .CURRENT_UCB[UCB$L_DEVDEPEND];
341 0722 2      SCRATCH = .HDR1 + SCRATCH_OFFSET;
342 0723 2      FLAGS = $FIELDMASK(MOUSV_REWIND);
343 0724 2
344 0725 2      ! This next call will use the UCB address to get the device's name and
345 0726 2      ! will fill in the fields with that name and the length of the name.
346 0727 2
347 0728 2      GET_DEV_NAME(CVT_DEVNAM_LENGTH,CVT_DEVNAM);
348 0729 2
349 0730 2      WHILE 1
350 0731 2      DO
351 0732 2          BEGIN
352 0733 2              WHILE 1
353 0734 2              DO
354 0735 2                  BEGIN
355 0736 2                      ! mount the volume, check if overwrite is possible
356 0737 2
357 0738 2                      MVL_ENTRY = MOUNT_VOL(.CURRENT_VCB[VCB$B_CUR_RVN], .FLAGS);
358 0739 2                      MVL = .CURRENT_VCB[VCB$L_MVL];
359 0740 2
360 0741 2                      ! set operator flag for "/INIT" and "/BLANK". If the operator
361 0742 2                      ! was required to intervene then these flags may be set.
362 0743 2
363 0744 2                      OPR_FLAG [ BLANK ] = (.MAIL [ OPC$W_MS_STATUS ] EQL
364 0745 2                      ( OPC$ BLANKTAPE AND %X'FFFF' ));
365 0746 2                      OPR_FLAG [ INIT ] = (.MAIL [ OPC$W_MS_STATUS ] EQL
366 0747 2                      ( OPC$_INITAPE AND %X'FFFF' ));
367 0748 2
368 0749 2                      ! do not check things on "/BLANK" or if the volume was mounted
369 0750 2                      ! "/BLANK"
370 0751 2
371 0752 2                      IF .OPR_FLAG[BLANK] OR .CURRENT_VCB[VCB$V_BLANK] THEN EXITLOOP;
372 0753 2
373 0754 2                      ! see if we can overwrite the 1st file, save the VOL1 access
374 0755 2                      ! character ( for defaulting ) before scratching the scratch area
375 0756 2
376 0757 2
377 0758 2
```

```

: 378      0759 4      ACCESS CHAR = .SCRATCH [ VL1$B VOLACCESS ];
: 379      0760 4      CH$MOVE (VL1$$ OWNER IDENT, SCRATCH[VL1$T OWNER IDENT],VOL_OWNER);
: 380      0761 5      ERROR_NO = CHECK_HDR(.MVL_ENTRY, (.OPR_FLAG[INIT]
: 381      0762 4      OR .CURRENT_VCB[VCB$V_INIT]));
: 382      0763 4
: 383      0764 4      ! check on the results
: 384      0765 4
: 385      0766 6      IF .ERROR_NO OR ((.OPR_FLAG[INIT] OR .CURRENT_VCB[VCB$V_INIT])
: 386      0767 5      AND (.ERROR_NO EQL MOUN$_NOTANSI))
: 387      0768 4      THEN EXITLOOP;
: 388      0769 4
: 389      0770 4      ! the tape is not ANSI without /INIT or /BLANK
: 390      0771 4
: 391      0772 4      PRINT_OPR_MSG(.ERROR_NO, 0,
: 392      0773 4      .CVT_DEVNAM_LENGTH,CVT_DEVNAM);
: 393      0774 4
: 394      0775 4      ! force physical mount
: 395      0776 4
: 396      0777 4      FLAGS = $FIELDMASK(MOU$V_REWIND) + $FIELDMASK(MOU$V_MOUNTERR);
: 397      0778 4      KERNEL_CALL(RESET_UNIT);
: 398      0779 3      END;
: 399      0780 3
: 400      0781 3      ERROR_NO = MOUN$_IOERROR;
: 401      0782 3
: 402      0783 3      ! try to initialize
: 403      0784 3
: 404      0785 3      IF REWIND_AND_WAIT()
: 405      0786 3      THEN
: 406      0787 4      BEGIN
: 407      0788 4
: 408      0789 4      ! fill with spaces
: 409      0790 4
: 410      0791 4      CH$FILL(' ', ANSI_LBLSZ, .SCRATCH);
: 411      0792 4      CH$FILL(' ', ANSI_LBLSZ, SCRATCH2);
: 412      0793 4
: 413      0794 4      ! Set defaults
: 414      0795 4
: 415      0796 4      .SCRATCH = 'VOL1';
: 416      0797 4      SCRATCH[VL1$B_LBL$STDVER] = .MVL[MVL$B_STDVER] + '0';
: 417      0798 4      SCRATCH2 = 'VOL2';
: 418      0799 4      (SCRATCH2[VL2$T_VOLOWNER]<0,32> = 'D%C ');
: 419      0800 4
: 420      0801 4      ! get the volume label from the MVL
: 421      0802 4
: 422      0803 4      CH$COPY(MVL$$_VOLLBL, MVL_ENTRY[MVL$T_VOLLBL], ' ',
: 423      0804 4      VL1$$_VOLLBL, SCRATCH[VL1$T_VOLLBL]);
: 424      0805 4
: 425      0806 4      ! If the operator supplied a label or if the MTAACP created
: 426      0807 4      ! the label, the ANSI volume owner from the MVL is stored in
: 427      0808 4      ! the label else the one currently on the tape will be used.
: 428      0809 4      ! The accessibility char to input to $MTACCESS is determined
: 429      0810 4      ! in a similar fashion, except it is not stored in the
: 430      0811 4      ! label until $MTACCESS has seen it.
: 431      0812 4
: 432      0813 4      IF (.MAILSZ NEQ 0) OR .OPR_FLAG [ INIT ] OR .OPR_FLAG [ BLANK ]
: 433      0814 4      OR .CURRENT_VCB [ VCB$V_INIT ]
: 434      0815 4      OR .CURRENT_VCB [ VCB$V_BLANK ]

```


435	0816	4
436	0817	5
437	0818	5
438	0819	5
439	0820	5
440	0821	5
441	0822	4
442	0823	5
443	0824	5
444	0825	5
445	0826	4
446	0827	4
447	0828	4
448	0829	4
449	0830	4
450	0831	4
451	0832	4
452	0833	4
453	0834	4
454	0835	4
455	0836	4
456	0837	4
457	0838	4
458	0839	4
459	0840	4
460	0841	4
461	0842	4
462	0843	4
463	0844	4
464	0845	4
465	0846	5
466	0847	5
467	0848	5
468	0849	5
469	0850	5
470	0851	5
471	0852	5
472	0853	5
473	0854	5
474	0855	5
475	0856	5
476	0857	5
477	0858	6
478	0859	6
479	0860	6
480	0861	6
481	0862	6
482	0863	5
483	0864	5
484	0865	5
485	0866	5
486	0867	5
487	0868	5
488	0869	5
489	0870	5
490	0871	5
491	0872	6

```

THEN
BEGIN
    ACCESS_CHAR = .MVL[MVL$B_VOL_ACC];
    CH$MOVE(VL1$$ OWNER_IDENT, MVL[MVL$T_VOLOWNER],
            SCRATCH[VL1$T_OWNER_IDENT]);
END
ELSE
BEGIN
    CH$MOVE (VL1$$ OWNER_IDENT, VOL_OWNER,
            SCRATCH [VC1$T_OWNER_IDENT]);
END;

! Call the accessibility system service to get the character to output.
! First keep the record that the UCB is reading. The accessibility
! routine can not move the tape from under us! Thus we will compare
! this to the field after the call and if the tape was moved we punt
! the operation.

ORB = .CURRENT_UCB[UCB$L_ORB];
CURRENT_RECORD = KERNEL_CALL (GET_RECORD, .CURRENT_UCB);
CHAR = $MTACCESS(LBLNAM = 0,
                UIC = .ORB[ORB$L_OWNER],
                STD_VERSION = .MVL[MVL$B_STDVER],
                ACCESS_CHAR = .ACCESS_CHAR,
                ACCESS_SPEC = MTASK_CHARVALID,
                TYPE = MTASK_OUTVOLT);

STATUS = KERNEL_CALL( GET_RECORD, .CURRENT_UCB);
IF .CURRENT_RECORD EQL .STATUS
THEN
BEGIN
    LOCAL    TMP_PROT          : WORD;          ! SOGW protection word

    ! Set the access char in the label
    SCRATCH[VL1$B_VOLACCESS] = .CHAR[0];

    ! fill in the VOL2 VMS owner field
    IF .ORB[ORB$V_PROT_16]
    THEN TMP_PROT = .ORB[ORB$W_PROT]
    ELSE
        BEGIN
            TMP_PROT<0,4> = .(ORB[ORB$L_SYS_PROT])<0,4>;
            TMP_PROT<4,4> = .(ORB[ORB$L_OWN_PROT])<0,4>;
            TMP_PROT<8,4> = .(ORB[ORB$L_GRP_PROT])<0,4>;
            TMP_PROT<12,4> = .(ORB[ORB$C_WOR_PROT])<0,4>;
        END;
    FORMAT_VOLOWNER(SCRATCH2, .ORB[ORB$L_OWNER], .TMP_PROT);

    ! If a VMS protection is specified and the user does not
    ! wish us to limit this to only ANSI standard only then
    ! write our system code in the VOL1 label. This will
    ! tell other implemenations that the VOL2 label on this
    ! tape was written by VMS.

    IF NOT (.CURRENT_VCB[VCB$V_INTCHG]

```

```

492      0873 6      AND .CURRENT VCB [VCB$V NOVOL2])
493      0874 6      AND (.ORB[ORB$$_SYS_PROT] NEQ 0 OR
494      0875 6      .ORB[ORB$$_OWN_PROT] NEQ 0 OR
495      0876 6      .ORB[ORB$$_GRP_PROT] NEQ 0 OR
496      0877 6      .ORB[ORB$$_WOR_PROT] NEQ 0)
497      0878 5      THEN
498      0879 6      BEGIN
499      0880 6      CH$MOVE(10,STARID,SCRATCH[VL1$T_SYSCODE]);
500      0881 6      SCRATCH[VL1$B_LBL$STDVER] = '4';
501      0882 5      END;
502      0883 5      ! set the same characteristics and if that succeeds write the
503      0884 5      ! label.
504      0885 5      IF ISSUE_IO(IOS$_SETMODE, SAVE_DEVCHAR, 0)
505      0886 5      THEN
506      0887 5      STATUS = ISSUE_IO(IOS$_WRITEBLK, .SCRATCH, ANSI_LBLSZ);
507      0888 5      ! If the frist write worked, then check to see if a VOL2 label needs
508      0889 5      ! to be written. If it does and that worked then exitloop.
509      0890 5      IF .STATUS
510      0891 5      THEN
511      0892 5      BEGIN
512      0893 5      IF NOT (.CURRENT VCB[VCB$V INTCHG]
513      0894 5      AND .CURRENT VCB [VCB$V NOVOL2])
514      0895 5      AND (.ORB[ORB$$_SYS_PROT] NEQ 0 OR
515      0896 6      .ORB[ORB$$_OWN_PROT] NEQ 0 OR
516      0897 7      .ORB[ORB$$_GRP_PROT] NEQ 0 OR
517      0898 7      .ORB[ORB$$_WOR_PROT] NEQ 0)
518      0899 7      THEN
519      0900 7      STATUS = ISSUE_IO (IOS$_WRITEBLK, SCRATCH2,
520      0901 7      ANSI_LBLSZ);
521      0902 7      IF .STATUS THEN EXITLOOP;
522      0903 6      END;
523      0904 6      IF .STATUS<0,16> EQL SS$_WRITLCK THEN ERROR_NO = MOUN$_WRITLCK;
524      0905 6      END
525      0906 6      ELSE
526      0907 5      ERROR_NO = MOUN$_TAPEPOSLOST;
527      0908 5      END;
528      0909 5      PRINT_OPR_MSG(.ERROR_NO, 0,
529      0910 5      .CVT_DEVNAM_LENGTH,CVT_DEVNAM);
530      0911 5      ! force physical mount
531      0912 5      FLAGS = $FIELDMASK(MOUSV_REWIND) + $FIELDMASK(MOUSV_MOUNTERR);
532      0913 5      KERNEL_CALL(RESET_UNIT);
533      0914 5      END;
534      0915 5      ! end of routine
535      0916 5      END;
536      0917 5
537      0918 5
538      0919 5
539      0920 5
540      0921 5
541      0922 5
542      0923 5
543      0924 5
544      0925 1
```


00 00 41 31 31 45 4C 49 46 43 45 44 000DC P.AAA: .ASCII \DECFILE11A\<0><0>

STARID= P.AAA
.EXTRN SYSSMTACCESS

5E FF68 CE 9E 00000 GTNEXT_VOL WRIT::

```
MOVAB -152(SP), SP
CLRL -(SP)
PUSHL SP
PUSHAB INC_VOL_SECTION
CALLS #3, @#SYSSCMKRNL
MOVL CURRENT_UCB, R0
MOVQ 64(R0), -SAVE_DEVCHAR
ADDL3 #320, HDR1, SCRATCH
MOVL #1, FLAGS
PUSHAB CVT_DEVNAM
PUSHAB CVT_DEVNAM_LENGTH
CALLS #2, GET_DEV_NAME
MOVZBL CVT_DEVNAM_LENGTH, 20(SP)
MOVZBL CVT_DEVNAM_LENGTH, 20(SP)
PUSHL FLAGS
MOVZBL 47(CURRENT_VCB), -(SP)
CALLS #2, MOUNT_VOL
MOVL R0, MVL_ENTRY
MOVL 52(CURRENT_VCB), MVL
CLRL R0
CMPW MAIL+2, #33251
BNEQ 2$
INCL R0
INSV R0, #0, #1, OPR_FLAG
CLRL R0
CMPW MAIL+2, #33235
BNEQ 3$
INCL R0
INSV R0, #1, #1, OPR_FLAG
BLBS OPR_FLAG, 6$
BBS #2, 45(CURRENT_VCB), 6$
MOVB 10(SCRATCH), ACCESS_CHAR
MOVCS #14, 37(SCRATCH), VOL_OWNER
EXTZV #1, #1, OPR_FLAG, R0
EXTZV #3, #1, 45(CURRENT_VCB), R1
BISL3 R1, R0, -(SP)
PUSHL MVL_ENTRY
BSBW CHECK_HDR
ADDL2 #8, SP
MOVL R0, ERROR_NO
BLBS ERROR_NO, 6$
BBS #1, OPR_FLAG, 5$
BBS #3, 45(CURRENT_VCB), 5$
BRW 22$
CMPL ERROR_NO, #7504124
BNEQ 4$
MOVL #7504164, ERROR_NO
CALLS #0, REWIND_AND_WAIT
BLBC R0, 4$
MOVCS #0, (SP), #32, #80, (SCRATCH)
```

```
0645
0719
0720
0722
0723
0728
0773
0917
0740
0741
0747
0746
0749
0748
0754
0759
0760
0762
0761
0766
0767
0781
0785
0791
```

0050 8F 20

0050	8F	20	6E	00	2C	000D8	MOVCS	#0, (SP), #32, #80, SCRATCH2	0792	
				AE		000DF				
			69	314C4F56	8F	D0	000E1	MOVL	#827084630, (SCRATCH)	0796
4F	A9	22	A7		30	81	000E8	ADDB3	#48, 34(MVL), 79(SCRATCH)	0797
		20	AE	324C4F56	8F	D0	000EE	MOVL	#843861846, SCRATCH2	0798
		24	AE	20432544	8F	D0	000F6	MOVL	#541271364, SCRATCH2+4	0799
04	A9	08	BE		06	28	000FE	MOVCS	#6, @MVL_ENTRY, 4(SCRATCH)	0804
				0000G	CF	D5	00104	TSTL	MAILSZ	0813
					11	12	00108	BNEQ	7\$	
	0D		5A		01	E0	0010A	BBS	#1, OPR_FLAG, 7\$	
			0A		5A	E8	0010E	BLBS	OPR_FLAG, 7\$	
	05	2D	AB		03	E0	00111	BBS	#3, 45(CURRENT_VCB), 7\$	0814
	0D	2D	AB		02	E1	00116	BBC	#2, 45(CURRENT_VCB), 8\$	0815
		0C	AE	12	A7	90	0011B	7\$:	18(MVL), ACCESS_CHAR	0818
25	A9	14	A7		0E	28	00120	MOVB	#14, 20(MVL), 37(SCRATCH)	0820
					06	11	00126	MOVCS	9\$	0813
25	A9	70	AE		0E	28	00128	8\$:	#14, VOL_OWNER, 37(SCRATCH)	0825
			50	0000G	CF	D0	0012E	9\$:	CURRENT_UCB, R0	0834
			56	1C	A0	D0	00133	MOVL	28(R0), -ORB	
					50	DD	00137	PUSHL	R0	0835
					01	DD	00139	PUSHL	#1	
				0000G	5E	DD	0013B	PUSHL	SP	
					CF	9F	0013D	PUSHAB	GET_RECORD	
		00000000G	9F		04	FB	00141	CALLS	#4, @#SYSSCMKRN	
		04	AE		50	D0	00148	MOVL	R0, CURRENT_RECORD	
					02	DD	0014C	PUSHL	#2	0841
					01	DD	0014E	PUSHL	#1	
			7E	14	AE	9A	00150	MOVZBL	ACCESS_CHAR, -(SP)	
			7E	22	A7	9A	00154	MOVZBL	34(MVL), -(SP)	
					66	DD	00158	PUSHL	(ORB)	
					7E	D4	0015A	CLRL	-(SP)	
		00000000G	00		06	FB	0015C	CALLS	#6, SYSSMTACCESS	
			6E		50	D0	00163	MOVL	R0, CHAR	
				0000G	CF	DD	00166	PUSHL	CURRENT_UCB	0843
					01	DD	0016A	PUSHL	#1	
				0000G	5E	DD	0016C	PUSHL	SP	
					CF	9F	0016E	PUSHAB	GET_RECORD	
		00000000G	9F		04	FB	00172	CALLS	#4, @#SYSSCMKRN	
		10	AE		50	D0	00179	MOVL	R0, STATUS	
		10	AE	04	AE	D1	0017D	CMPL	CURRENT_RECORD, STATUS	0844
					03	13	00182	BEQL	10\$	
				00C9	31	00184	BRW	21\$		
		0A	A9		6E	90	00187	10\$:	CHAR, 10(SCRATCH)	0851
			06	08	A6	E9	0018B	MOVB	11(ORB), 11\$	0855
			50	18	A6	B0	0018F	BLBC	24(ORB), TMP_PROT	0856
					18	11	00193	MOVW	12\$	
					A6	F0	00195	BRB	24(ORB), #0, #4, TMP_PROT	0859
50		04	00	18	A6	F0	0019B	INSV	28(ORB), #4, #4, TMP_PROT	0860
50		04	04	1C	A6	F0	001A1	INSV	32(ORB), #8, #4, TMP_PROT	0861
50		04	08	20	A6	F0	001A7	INSV	36(ORB), #12, #4, TMP_PROT	0862
50		04	0C	24	A6	F0	001AD	INSV	TMP_PROT, -(SP)	0864
			7E		50	3C	001B0	MOVZWL	(ORB)	
					66	DD	001B2	PUSHL	SCRATCH2	
				28	AE	9F	001B5	PUSHAB	#3, FORMAT_VOLOWNER	
		0000G	CF		03	FB	001B5	CALLS	#4, 44(CURRENT_VCB), 13\$	0872
	05	2C	AB		04	E1	001BA	BBC	#6, 44(CURRENT_VCB), 15\$	0873
	1F	2C	AB		06	E0	001BF	BBS	24(ORB)	0874
				18	A6	D5	001C4	TSTL		

18	A9	FE17	CF	1C	0F	12	001C7	BNEQ	14\$	0875		
		4F	A9		A6	D5	001C9	TSTL	28(ORB)	0876		
				20	0A	12	001CC	BNEQ	14\$	0877		
					A6	D5	001CE	TSTL	32(ORB)	0880		
				24	05	12	001D1	BNEQ	14\$	0881		
					A6	D5	001D3	TSTL	36(ORB)	0887		
					0B	13	001D6	BEQL	15\$			
					0A	28	001D8	14\$: MOV C3	#10, STARID, 24(SCRATCH)			
					34	90	001DF	MOV B	#52, 79(SCRATCH)			
					7E	D4	001E3	15\$: CLRL	-(SP)			
				0084	CE	9F	001E5	PUSHAB	SAVE_DEVCHAR			
					23	DD	001E9	PUSHL	#35			
					0000G	30	001EB	BSBW	ISSUE IO			
					0C	C0	001EE	ADDL2	#12, SP			
					50	E9	001F1	BLBC	R0, 16\$			
				50	8F	9A	001F4	MOVZBL	#80, -(SP)	0889		
					59	DD	001F8	PUSHL	SCRATCH			
					20	DD	001FA	PUSHL	#32			
					0000G	30	001FC	BSBW	ISSUE IO			
					0C	C0	001FF	ADDL2	#12, SP			
					50	D0	00202	MOVL	R0, STATUS			
				10	AE	E9	00206	16\$: BLBC	STATUS, 20\$	0894		
					04	E1	0020A	BBC	#4, 44(CURRENT_VCB), 17\$	0897		
					06	E0	0020F	BBS	#6, 44(CURRENT_VCB), 19\$	0898		
				18	A6	D5	00214	17\$: TSTL	24(ORB)	0899		
					0F	12	00217	BNEQ	18\$			
				1C	A6	D5	00219	TSTL	28(ORB)	0900		
					0A	12	0021C	BNEQ	18\$			
				20	A6	D5	0021E	TSTL	32(ORB)	0901		
					05	12	00221	BNEQ	18\$			
				24	A6	D5	00223	TSTL	36(ORB)	0902		
					13	13	00226	BEQL	19\$			
					7E	8F	9A	00228	18\$: MOVZBL	#80, -(SP)	0904	
				50	AE	9F	0022C	PUSHAB	SCRATCH2			
				24	20	DD	0022F	PUSHL	#32			
					0000G	30	00231	BSBW	ISSUE IO			
					0C	C0	00234	ADDL2	#12, SP			
					50	D0	00237	MOVL	R0, STATUS			
				10	AE	E8	0023B	19\$: BLBS	STATUS, 23\$	0906		
					10	AE	B1	0023F	20\$: CMPW	STATUS, #604	0910	
					10	12	00245	BNEQ	22\$			
					58	00728134	8F	D0	00247	MOVL	#7504180, ERROR_NO	
					07	11	0024E	BRB	22\$	0844		
					58	00728274	8F	D0	00250	21\$: MOVL	#7504500, ERROR_NO	0914
					0088	CE	9F	00257	22\$: PUSHAB	CVT_DEVNAM	0916	
					18	AE	DD	0025B	PUSHL	24(SP)	0917	
						7E	D4	0025E	CLRL	-(SP)	0916	
						58	DD	00260	PUSHL	ERROR_NO		
					0000G	30	00262	BSBW	PRINT_OPR_MSG			
					10	C0	00265	ADDL2	#16, SP			
					18	AE	09	D0	00268	MOVL	#9, FLAGS	0921
						7E	D4	0026C	CLRL	-(SP)	0922	
						5E	DD	0026E	PUSHL	SP		
					0000V	CF	9F	00270	PUSHAB	RESET UNIT		
						03	FB	00274	CALLS	#3, @SYSSCMKRNL		
						FDC5	31	0027B	BRW	1\$	0730	
						CE	9E	0027E	23\$: MOVAB	152(SP), SP	0925	

NXTVOL
V04-000

L 14
16-Sep-1984 02:27:10
14-Sep-1984 12:46:45

VAX-11 Bliss-32 V4.0-742
[MTAACP.SRC]NXTVOL.B32;1

Page 16
(3)

05 00283

RSB

;

; Routine Size: 644 bytes, Routine Base: \$CODE\$ + 00E8

; 545 0926 1


```

547 0927 1 ROUTINE INC_VOL_SECTION : COMMON_CALL NOVALUE =
548 0928 1
549 0929 1 ++
550 0930 1
551 0931 1 FUNCTIONAL DESCRIPTION:
552 0932 1 This routine increments the relative volume number
553 0933 1 and the file section number
554 0934 1
555 0935 1 CALLING SEQUENCE:
556 0936 1 INC_VOL_SECTION(), CALLED IN KERNEL MODE
557 0937 1
558 0938 1 INPUT PARAMETERS:
559 0939 1 NONE
560 0940 1
561 0941 1 IMPLICIT INPUTS:
562 0942 1 CURRENT_VCB - address of volume control block
563 0943 1
564 0944 1 OUTPUT PARAMETERS:
565 0945 1 NONE
566 0946 1
567 0947 1 IMPLICIT OUTPUTS:
568 0948 1 CURRENT_VCB[VCBSB_CUR_RVN] incremented
569 0949 1 CURRENT_VCB[VCBSW_CUR_SEQ] incremented
570 0950 1
571 0951 1 ROUTINE VALUE:
572 0952 1 NONE
573 0953 1
574 0954 1 SIDE EFFECTS:
575 0955 1 NONE
576 0956 1
577 0957 1 --
578 0958 1
579 0959 2 BEGIN
580 0960 2
581 0961 2 EXTERNAL REGISTER
582 0962 2 COMMON_REG;
583 0963 2
584 0964 2 CURRENT_VCB[VCBSB_CUR_RVN] = .CURRENT_VCB[VCBSB_CUR_RVN] + 1;
585 0965 2 CURRENT_VCB[VCBSW_CUR_SEQ] = .CURRENT_VCB[VCBSW_CUR_SEQ] + 1;
586 0966 2 CURRENT_VCB[VCBSB_TM] = 0;
587 0967 2 CURRENT_VCB[VCBSL_ST_RECORD] = 0;
588 0968 1 END;

```

! end of routine

0000 00000 INC_VOL_SECTION:

2F	AB	96	00002	.WORD	Save nothing
26	AB	86	00005	INCB	47(CURRENT_VCB)
2E	AB	94	00008	INCW	38(CURRENT_VCB)
30	AB	D4	0000B	CLRB	46(CURRENT_VCB)
		04	0000E	CLRL	48(CURRENT_VCB)
				RET	

0927
0964
0965
0966
0967
0968

; Routine Size: 15 bytes, Routine Base: \$CODE\$ + 036C

NXTVOL
V04-000

N 14
16-Sep-1984 02:27:10
14-Sep-1984 12:46:45

VAX-11 Bliss-32 V4.0-742
[MTAACP.SRC]NXTVOL.B32;1

Page 18
(4)

OP
VO


```

: 590 0969 1 ROUTINE UPDATE_MVL_LBL (MVL_ENTRY, ADDR) : COMMON_CALL NOVALUE =
: 591 0970 1
: 592 0971 1 ++
: 593 0972 1
: 594 0973 1 FUNCTIONAL DESCRIPTION:
: 595 0974 1 This routine updates the relative volume label from the vol1 label
: 596 0975 1
: 597 0976 1 CALLING SEQUENCE:
: 598 0977 1 UPDATE_MVL_LBL(ARG1,ARG2)
: 599 0978 1
: 600 0979 1 INPUT PARAMETERS:
: 601 0980 1 ARG1 - address of mvl entry for current volume
: 602 0981 1 ARG2 - address of volume label on this tape volume
: 603 0982 1
: 604 0983 1 IMPLICIT INPUTS:
: 605 0984 1 NONE
: 606 0985 1
: 607 0986 1 OUTPUT PARAMETERS:
: 608 0987 1 NONE
: 609 0988 1
: 610 0989 1 IMPLICIT OUTPUTS:
: 611 0990 1 NONE
: 612 0991 1
: 613 0992 1 ROUTINE VALUE:
: 614 0993 1 NONE
: 615 0994 1
: 616 0995 1 SIDE EFFECTS:
: 617 0996 1 NONE
: 618 0997 1
: 619 0998 1 USER ERRORS:
: 620 0999 1 NONE
: 621 1000 1
: 622 1001 1 --
: 623 1002 1
: 624 1003 2 BEGIN
: 625 1004 2
: 626 1005 2 EXTERNAL REGISTER COMMON_REG;
: 627 1006 2
: 628 1007 2 EXTERNAL
: 629 1008 2 ANSI_A_GOOD : VECTOR [ , BYTE ];! translation table for ANSI 'a' char
: 630 1009 2
: 631 1010 2 MAP
: 632 1011 2 MVL_ENTRY : REF BBLOCK;
: 633 1012 2
: 634 1013 2 ! translate the label into upper case and put in ' ' for any non-ANSI
: 635 1014 2 ! 'a' characters found
: 636 1015 2
: 637 1016 2 CH$TRANSLATE (ANSI_A_GOOD, VL1$$_VOLLBL, ADDR, ' '
: 638 1017 2 MVL$$_VOLLBL, MVL_ENTRY [MVL$_VOLLBL] );
: 639 1018 2 MVL_ENTRY [ MVL$_UNUSED ] = 0;
: 640 1019 1 END;
```

.EXTRN ANSI_A_GOOD

007C 00000 UPDATE_MVL_LBL:

NXTVOL
V04-000

C 15
16-Sep-1984 02:27:10
14-Sep-1984 12:46:45

VAX-11 Bliss-32 V4.0-742
[MTAACP.SRC]NXTVOL.B32;1

Page 20
(5)

0000G	CF	20	08	56	04	AC	D0	00002
				BC		06	2E	00006
				66		06		0000E
			07	A6		02	8A	00010
						04		00014

.WORD	Save R2,R3,R4,R5,R6
MOVL	MVL_ENTRY, R6
MOVTC	#6, @ADDR, #32, ANSI_A_GOOD, #6, (R6)
BICB2	#2, 7(R6)
RET	

:	0969
:	1017
:	
:	
:	1018
:	1019

; Routine Size: 21 bytes, Routine Base: \$CODE\$ + 037B


```

: 642      1020 1 ROUTINE CHECK_HDR ( MVL_ENTRY, SLASH_INIT ) : L$CHECK_HDR =
: 643      1021 1
: 644      1022 1 ++
: 645      1023 1
: 646      1024 1 FUNCTIONAL DESCRIPTION:
: 647      1025 1     This routine checks that the tape can be overwritten.
: 648      1026 1
: 649      1027 1 CALLING SEQUENCE:
: 650      1028 1     CHECK_HDR(ARG1,ARG2)
: 651      1029 1
: 652      1030 1 INPUT PARAMETERS:
: 653      1031 1     ARG1 - address of current mounted volume entry
: 654      1032 1     ARG2 - is this a "/INIT"
: 655      1033 1
: 656      1034 1 IMPLICIT INPUTS:
: 657      1035 1     NONE
: 658      1036 1
: 659      1037 1 OUTPUT PARAMETERS:
: 660      1038 1     NONE
: 661      1039 1
: 662      1040 1 IMPLICIT OUTPUTS:
: 663      1041 1     NONE
: 664      1042 1
: 665      1043 1 ROUTINE VALUE:
: 666      1044 1     1 - ok to write
: 667      1045 1     various error codes
: 668      1046 1
: 669      1047 1 SIDE EFFECTS:
: 670      1048 1     NONE
: 671      1049 1
: 672      1050 1 --
: 673      1051 1
: 674      1052 2 BEGIN
: 675      1053 2
: 676      1054 2 MAP
: 677      1055 2     MVL_ENTRY      : REF BBLOCK;
: 678      1056 2
: 679      1057 2 EXTERNAL REGISTER
: 680      1058 2     SCRATCH = 9      : REF BBLOCK,
: 681      1059 2     COMMON_REG;
: 682      1060 2
: 683      1061 2 BIND
: 684      1062 2     USER_VOL_LABEL = UPLIT ( 'UVL' ),      ! user's volume labels code
: 685      1063 2     VOLUME_LABEL  = UPLIT ( 'VOL' );      ! other volume labels code
: 686      1064 2
: 687      1065 2 LOCAL
: 688      1066 2     MVL      : REF BBLOCK,      ! MVL address
: 689      1067 2     ORB      : REF BBLOCK,      ! ORB address
: 690      1068 2     STATUS,
: 691      1069 2     CURRENT_RECORD,      ! curr record drive is reading
: 692      1070 2     ACCESS;      ! Users' access to overwrite tape
: 693      1071 2
: 694      1072 2 ! loop till we find HDR1
: 695      1073 2
: 696      1074 2 WHILE 1
: 697      1075 2 DO
: 698      1076 3     BEGIN
```

```

: 699      1077      3
: 700      1078      3
: 701      1079      3
: 702      1080      3
: 703      1081      3
: 704      1082      3
: 705      1083      3
: 706      1084      3
: 707      1085      3
: 708      1086      3
: 709      1087      3
: 710      1088      3
: 711      1089      3
: 712      1090      3
: 713      1091      3
: 714      1092      3
: 715      1093      3
: 716      1094      3
: 717      1095      3
: 718      1096      3
: 719      1097      3
: 720      1098      3
: 721      1099      3
: 722      1100      3
: 723      1101      3
: 724      1102      3
: 725      1103      3
: 726      1104      3
: 727      1105      3
: 728      1106      3
: 729      1107      3
: 730      1108      3
: 731      1109      3
: 732      1110      3
: 733      1111      3
: 734      1112      3
: 735      1113      3
: 736      1114      3
: 737      1115      3
: 738      1116      3
: 739      1117      3
: 740      1118      3
: 741      1119      3
: 742      1120      3
: 743      1121      3
: 744      1122      3
: 745      1123      3
: 746      1124      3
: 747      1125      3
: 748      1126      3
: 749      1127      3
: 750      1128      3
: 751      1129      3
: 752      1130      3
: 753      1131      3
: 754      1132      3
: 755      1133      3

STATUS = ISSUE_IO( IOS_READBLK, .SCRATCH, ANSI_LBLSZ);
IF (.STATUS<0,16> EQL SSS_ENDOFFILE) AND .SLASH_INIT
THEN RETURN TRUE;

IF (NOT .STATUS) AND (.STATUS<0,16> NEQ SSS_DATAOVERUN)
THEN RETURN MOUN$_IOERROR;

IF .(.SCRATCH) EQL 'HDR1' THEN EXITLOOP;

! if we do not see a valid member of the volume label group THEN FAIL
IF NOT ( ( CH$EQL ( 3, .SCRATCH, 3, USER_VOL_LABEL ))
        OR ( CH$EQL ( 3, .SCRATCH, 3, VOLUME_LABEL ))
        )
THEN RETURN MOUN$_NOTANSI;

END;

! Call the accessibility system service to check the accessibility char
! on the HDR1 label.
! First keep the record that the UCB is reading. The accessibility
! routine can not move the tape from under us! Thus we will compare
! this to the field after the call and if the tape was moved we punt
! the operation. The check the code return from the system service
! to determine what type of access the user was granted.

MVL = .CURRENT_VCB[VCB$_MVL];
ORB = .CURRENT_UCB[UCB$_ORB];
CURRENT_RECORD = KERNEL CALL(GET_RECORD, .CURRENT_UCB);
ACCESS = $MTACCESS(LBLNAM = .SCRATCH,
                   UIC = .ORB[ORB$_OWNER],
                   STD_VERSION = .MVL[MVL$_STDVER],
                   ACCESS_CHAR = 0,
                   ACCESS_SPEC = MTASK_NOCHAR,
                   TYPE = MTASK_INHDR1);

STATUS = KERNEL CALL(GET_RECORD, .CURRENT_UCB);
IF .CURRENT_RECORD NEQ .STATUS
THEN RETURN (MOUN$_TAPEPOSLOST);

IF .ACCESS EQL SSS_FILACCERR
THEN
BEGIN
  IF NOT (.CURRENT_VCB[VCB$_OVRACC] AND .MVL_ENTRY [ MVL$_OVERRIDE ])
  THEN RETURN MOUN$_ACCERR;
  ACCESS = SSS_NORMAL;
END;

IF .ACCESS EQL SSS_NOVOLACC
THEN RETURN MOUN$_NOVOLACC;

IF .ACCESS EQL SSS_NOFILACC
THEN RETURN MOUN$_NOFILACC;

IF NOT ( ( .CURRENT_VCB[VCB$_OVREXP] AND .MVL_ENTRY [ MVL$_OVERRIDE ])
        OR
```


756 1134 3
757 1135 3
758 1136 2
759 1137 2
760 1138 2
761 1139 2
762 1140 1

EXPIRED (SCRATCH[HD1\$T_EXPIREDT])
)
THEN RETURN MOUN\$_FILNOTEXP;
RETURN TRUE;
END;

! end of routine CHECK_HDR

00 4C 56 55 00390 P.AAB: .ASCII \UVL\<0>
00 4C 4F 56 00394 P.AAC: .ASCII \VOL\<0>

USER_VOL_LABEL= P.AAB
VOLUME_LABEL= P.AAC

			3C	BB	00000	CHECK_HDR:		
	7E	50	8F	9A	00002	1\$: PUSH	#^M<R2,R3,R4,R5>	1020
			59	DD	00006	MOVZBL	#80, -(SP)	1078
			21	DD	00008	PUSHL	SCRATCH	
		0000G	30	0000A	PUSHL	#33		
	5E		0C	C0	0000D	BSBW	ISSUE_10	
	54		50	D0	00010	ADDL2	#12, SP	
0870	8F		54	B1	00013	MOVL	R0, STATUS	
			07	12	00018	CMPW	STATUS, #2160	1080
	03	18	AE	E9	0001A	BNEQ	2\$	
			00F6	31	0001E	BLBC	SLASH_INIT, 2\$	
	10		54	E8	00021	BRW	14\$	
0838	8F		54	B1	00024	2\$: BLBS	STATUS, 3\$	1083
			09	13	00029	CMPW	STATUS, #2104	
	50	00728124	8F	D0	0002B	BEQL	3\$	
			79	11	00032	MOVL	#7504164, R0	1084
31524448	8F		69	D1	00034	3\$: BRB	5\$	1086
			17	13	0003B	CMPL	(SCRATCH), #827475016	
B6 AF	69		03	29	0003D	BEQL	4\$	1090
			BE	13	00042	CMPC3	#3, (SCRATCH), USER_VOL_LABEL	
B3 AF	69		03	29	00044	BEQL	1\$	1091
			B7	13	00049	CMPC3	#3, (SCRATCH), VOLUME_LABEL	
	50	007280FC	8F	D0	0004B	BEQL	1\$	1093
			79	11	00052	MOVL	#7504124, R0	
52		34	AB	D0	00054	4\$: BRB	8\$	1105
50		0000G	CF	D0	00058	MOVL	52(CURRENT VCB), MVL	1106
53		1C	A0	D0	0005D	MOVL	CURRENT_UCB, R0	
			50	DD	00061	MOVL	28(R0), ORB	
			01	DD	00063	PUSHL	R0	1107
			5E	DD	00065	PUSHL	#1	
		0000G	CF	9F	00067	PUSHL	SP	
00000000G	9F		04	FB	0006B	PUSHAB	GET_RECORD	
55			50	D0	00072	CALLS	#4, @#SYSS\$CMKRNL	
			01	DD	00075	MOVL	R0, CURRENT_RECORD	
			7E	7C	00077	PUSHL	#1	1113
	7E	22	A2	9A	00079	CLRQ	-(SP)	
			63	DD	0007D	MOVZBL	34(MVL), -(SP)	
			59	DD	0007F	PUSHL	(ORB)	
00000000G	00		06	FB	00081	PUSHL	SCRATCH	
						CALLS	#6, SYSS\$MTACCESS	

	53		0000G	50	D0	00088	MOVL	R0, ACCESS		
				CF	DD	0008B	PUSHL	CURRENT_UCB	1114	
				01	DD	0008F	PUSHL	#1		
				5E	DD	00091	PUSHL	SP		
			0000G	CF	9F	00093	PUSHAB	GET_RECORD		
	00000000G	9F		04	FB	00097	CALLS	#4, @#SYSSCMKRN		
		54		50	D0	0009E	MOVL	R0, STATUS		
		54		55	D1	000A1	CMPL	CURRENT_RECORD, STATUS	1115	
				09	13	000A4	BEQL	6\$		
		50	00728274	8F	D0	000A6	MOVL	#7504500, R0	1116	
				6B	11	000AD	BRB	15\$		
	0000009C	8F		53	D1	000AF	CMPL	ACCESS, #156	1118	
				1A	12	000B6	BNEQ	10\$		
09	2C	AB		01	E1	000B8	BBC	#1, 44(CURRENT_VCB), 7\$	1121	
		50	14	AE	D0	000BD	MOVL	MVL_ENTRY, R0		
09	07	A0		02	E0	000C1	BBS	#2, 7(R0), 9\$		
		50	007280E4	8F	D0	000C6	MOVL	#7504100, R0	1122	
				4B	11	000CD	BRB	15\$		
		53		01	D0	000CF	MOVL	#1, ACCESS	1123	
	000022A4	8F		53	D1	000D2	CMPL	ACCESS, #8868	1126	
				09	12	000D9	BNEQ	11\$		
		50	00728264	8F	D0	000DB	MOVL	#7504484, R0	1127	
				36	11	000E2	BRB	15\$		
	000022AC	8F		53	D1	000E4	CMPL	ACCESS, #8876	1129	
				09	12	000EB	BNEQ	12\$		
		50	0072826C	8F	D0	000ED	MOVL	#7504492, R0	1130	
				24	11	000F4	BRB	15\$		
		09	2C	AB	E9	000F6	BLBC	44(CURRENT_VCB), 13\$	1132	
		50	14	AE	D0	000FA	MOVL	MVL_ENTRY, R0		
14	07	A0		02	E0	000FE	BBS	#2, 7(R0), 14\$		
			2F	A9	9F	00103	PUSHAB	47(SCRATCH)	1134	
	0000G	CF		01	FB	00106	CALLS	#1, EXPIRED		
		09		50	E8	0010B	BLBS	R0, 14\$		
		50	007280EC	8F	D0	0010E	MOVL	#7504108, R0	1136	
				03	11	00115	BRB	15\$		
		50		01	D0	00117	MOVL	#1, R0	1138	
				3C	BA	0011A	POPR	#^M<R2,R3,R4,R5>	1140	
				05	0011C		RSB			

; Routine Size: 285 bytes, Routine Base: \$CODE\$ + 0398

; 763 1141 1


```

: 765      1142 1 GLOBAL ROUTINE RESET_UNIT : COMMON_CALL NOVALUE =
: 766      1143 1
: 767      1144 1 ++
: 768      1145 1
: 769      1146 1 FUNCTIONAL DESCRIPTION:
: 770      1147 1
: 771      1148 1     This routine resets the unit so that after an error message
: 772      1149 1     the same unit is chosen for mount
: 773      1150 1
: 774      1151 1
: 775      1152 1 CALLING SEQUENCE:
: 776      1153 1
: 777      1154 1 INPUT PARAMETERS:
: 778      1155 1     NONE
: 779      1156 1
: 780      1157 1 IMPLICIT INPUTS:
: 781      1158 1     NONE
: 782      1159 1
: 783      1160 1 OUTPUT PARAMETERS:
: 784      1161 1     NONE
: 785      1162 1
: 786      1163 1 IMPLICIT OUTPUTS:
: 787      1164 1     NONE
: 788      1165 1
: 789      1166 1 ROUTINE VALUE:
: 790      1167 1     NONE
: 791      1168 1
: 792      1169 1 SIDE EFFECTS:
: 793      1170 1     NONE
: 794      1171 1
: 795      1172 1 --
: 796      1173 1
: 797      1174 2 BEGIN
: 798      1175 2
: 799      1176 2 EXTERNAL REGISTER
: 800      1177 2     COMMON_REG;
: 801      1178 2
: 802      1179 2 IF .CURRENT_VCB[VCB$W_RVN] NEQ 0
: 803      1180 2 THEN
: 804      1181 2     CURRENT_VCB[VCB$W_RVN] = .CURRENT_VCB[VCB$W_RVN] - 1
: 805      1182 2 ELSE
: 806      1183 2     CURRENT_VCB[VCB$W_RVN] = .BLOCK[.CURRENT_VCB[VCB$L_RVT], RVT$B_NVOLS]
: 807      1184 2     - 1;
: 808      1185 2
: 809      1186 1 END;

```

			0000 00000	.ENTRY	RESET UNIT, Save nothing	: 1142
		OE	AB B5 00002	ISTW	14(CURRENT_VCB)	: 1179
			09 12 00005	BNEQ	1\$	
	50	20	AB D0 00007	MOVL	32(CURRENT_VCB), R0	: 1183
OE	AB	0B	AO 9B 0000B	MOVZBW	11(R0), 14(CURRENT_VCB)	: 1184
		OE	AB B7 00010 1\$:	DECW	14(CURRENT_VCB)	: 1186
			04 00013	RET		

NXTVOL
V04-000

I 15
16-Sep-1984 02:27:10
14-Sep-1984 12:46:45

VAX-11 Bliss-32 V4.0-742
[MTAACP.SRC]NXTVOL.B32;1

Page 26
(7)

; Routine Size: 20 bytes, Routine Base: \$CODE\$ + 04B5

```
: 810      1187 1
: 811      1188 1 END
: 812      1189 1
: 813      1190 0 ELUDOM
```

PSECT SUMMARY

```
:
: Name          Bytes          Attributes
: $CODE$        1225 NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
```

Library Statistics

```
:
: File          Total Symbols Loaded Percent Pages Mapped Processing Time
: _$255$DUA28:[SYSLIB]LIB.L32;1 18619      82      0      1000      00:01.8
```

COMMAND QUALIFIERS

; BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:NXTVOL/OBJ=OBJ\$:NXTVOL MSRC\$:NXTVOL/UPDATE=(ENH\$:NXTVOL)

```
: Size:          1204 code + 21 data bytes
: Run Time:      00:24.7
: Elapsed Time:  00:53.4
: Lines/CPU Min: 2891
: Lexemes/CPU-Min: 19594
: Memory Used: 252 pages
: Compilation Complete
```


0255 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

